On Link Estimation in Dense RPL Deployments

Sébastien Dawans
CETIC
Rue des Frères Wright 29/3
B-6041 Charleroi, Belgium
sebastien.dawans@cetic.be

Simon Duquennoy
Swedish Institute of Computer Science (SICS)
SE-16429 Kista Stockholm, Sweden
simonduq@sics.se

Olivier Bonaventure
ICTEAM
Université Catholique de Louvain
B-1348 Louvain-la-Neuve, Belgium
olivier.bonaventure@uclouvain.be

***Abstract*—The Internet of Things vision foresees billions of devices to connect the physical world to the digital world. Sensing applications such as structural health monitoring, surveillance or smart buildings employ multi-hop wireless networks with high density to attain sufficient area coverage. Such applications need networking stacks and routing protocols that can scale with network size and density while remaining energy-efficient and lightweight. To this end, the IETF RoLL working group has designed the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL). This paper discusses the problems of link quality estimation and neighbor management policies when it comes to handling high densities. We implement and evaluate different neighbor management policies and link probing techniques in Contiki's RPL implementation. We report on our experience with a 100-node testbed with average 40-degree density. We show the sensitivity of high density routing with respect to cache sizes and routing metric initialization. Finally, we devise guidelines for design and implementation of density-scalable routing protocols.**

***Index Terms*—Sensor networks, Routing, Scalability, RPL**

I. Introduction

The emerging Internet of Things (IoT) envisions large networks of sensors and actuators to support various applications. For communication to happen in such networks, one needs routing protocols to be reliable, energy efficient, and most importantly, scalable. A good routing protocol in this context should scale with respect to both network size and density, in spite of the resource constraints of Wireless Sensor Networks (WSN) [6]

The research community has done a great job designing scalable and reliable routing solutions for sensor networks. The Collection Tree Protocol, CTP [3], is now a well-acknowledged solution for data collection in WSN. It informed the design of the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL), recently standardized by the IETF [14]. RPL makes CTP IPv6-ready, more flexible, and extends it with the ability to support any-to-any traffic.

This position paper highlights and discusses the problem of making routing scale to high-density networks. We show why and how high-density networks are challenging, in terms of next hop selection, neighbor information management, link discovery and link quality estimation. We propose potential solutions and discuss the use of active and passive probing, aiming to maintain a precise view of the surrounding links' quality, for better routing decisions.

We implement our solutions by extending RPL in the Contiki OS, and perform experiments in a large scale testbed with high density, TWIST [5]. Our results are encouraging, demonstrating the beneficial impact of our solutions, especially for links with intermediate quality.

The remainder of this paper is structured as follows. We start in Section II by giving background on the RPL protocol. Then, we introduce the problem of routing in dense networks and potential solutions in Section III. Section IV presents our preliminary results, obtained from experiments in the TWIST testbed. Finally, we discuss related work in Section V and conclude in Section VI.

II. Background

RPL addresses the problem of scalable routing in low-power IPv6 networks. In a few words, RPL performs distance-vector routing over a simple tree-like topology. The topology used is a Destination-Oriented Directed Acyclic Graph (DODAG), meaning that each node may have more than one parent towards the root. One of the parents is called *preferred parent*, and is used for routing towards the root.

The root acts as a sink and a gateway bridging the local network and the Internet. RPL provides good scalability with the network size because the entirety of the control traffic is focused in optimizing a single topology towards a single direction, rather than one route for each pair of nodes in the network. In order for RPL to construct viable DODAGs, each node needs to store information about its own neighborhood to allow the estimation of routing metrics and hop-forwarding costs.

RPL can be used with different routing metrics, defining how path costs are evaluated. This paper focuses on the most widely used and well-acknowledged Estimated Transmission Count (ETX) metric [1], which estimates the total number of transmissions needed to reach the destination. ETX is obtained by summing the inverse of the Packet Reception Rates (PRR) of each hop in the path.

The distance from a given node to the sink according to one or more metrics is called rank, and is calculated according to an Objective Function (OF). The objective function we focus on is the Minimum Rank with Hysteresis Objective Function (MRHOF), which supports any additive routing metric and uses ETX by default. Each node broadcasts its rank periodically, using a timer with exponential period (Trickle timer [10]). The path cost associated to a given neighbor is the sum of its rank and the ETX of the link towards it.

978-1-4673-2129-7/12/$31.00 ©2012 IEEE

RPL is highly configurable; For more details, we refer the reader to RFC 6550 [14].

III. Routing in Dense Networks

In certain IoT application domains, such as smart metering, smart cities or smart buildings, hundreds of devices may be deployed closely [11].

Dense networks are challenging from a wireless/MAC perspective, because they cause channel contention. Dense networks are also challenging from a routing perspective, because they offer many routing options, with heterogeneous and variable link qualities. Although ETX-based routing as performed by CTP or RPL is now well-acknowledged [3], [7], we argue that the problem of scaling with network density has not been analysed sufficiently by the community.

The next subsections review the challenges of routing in dense networks which we address.

A. Neighbor Discovery and Link Estimation

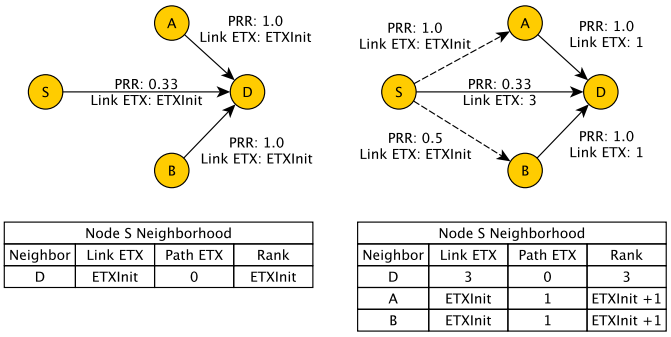
The basic component of an ETX-based routing topology is the individual quality of links. In dense networks, where nodes have many neighbors, estimating link qualities becomes difficult. In CTP and RPL, the quality of links is measured through datapath validation, i.e. by recording statistics on success and failures for the links used by data traffic. The ETX calculation is implemented as an Exponentially Weighted Moving Average (EWMA) filter.

This approach is conservative in the sense that it only evaluates the links that are currently being used. It efficiently detects link failures towards the current preferred parent, but doesn't investigate any alternative otherwise. As a result, CTP and RPL often stick to a routing topology that may become sub-optimal with time.

We observed this problem even in a stable network, where, as illustrated by Figure 1, a node keeps sending directly to the DODAG root – the first neighbor it had heard of – instead of investigating other neighbors. Although data packets require on average three transmissions to reach the sink in this example, alternative neighbors offering potentially more reliable paths – shown by dashed lines such as through node A which would result in a Path ETX of 2 – are not tested due to a high $EtxInit$, whose default values are reported to be 3.5 and 5.0 in two widely used RPL implementations [7].

A solution to the problem of link estimation is to actively probe the links by sending unicast messages to some of the available neighbors. Active probing might be periodic or event-base such as to test newly-discovered neighbors in order to improve the decision-making process when considering addition of new candidate neighbors to saturated neighbor tables.

Proposed Solution: To make probing scale with density, we propose an alternative approach, *passive probing*. Passive probing consists in having an extremely non-conservative preferred parent solution, so the next hop changes often, and all potentially good neighbors are eventually considered. To this end, we use optimistic link estimation, i.e. we assume that any unknown link is good (link ETX initialized to 1, the lowest possible value), combined with a low MRHOF hysteresis. By doing this, a rotation among good next hop candidates will happen. Datapath validation will operate on each candidate, and sort them by path ETX.



Node S Neighborhood			
Neighbor	Link ETX	Path ETX	Rank
D	ETXInit	0	ETXInit

Node S Neighborhood			
Neighbor	Link ETX	Path ETX	Rank
D	3	0	3
A	ETXInit	1	ETXInit +1
B	ETXInit	1	ETXInit +1

Fig. 1. Neighborhood information of sender S, to reach destination D. At bootstrap (left), nodes select an initial path toward D, with Link ETX of $ETXInit$. After convergence (right), $link_{S,D} = 3$, and alternate paths are not selected when $ETXinit$ is set to 2 or above.

With passive probing, a node having many neighbors will start by choosing the one with the best advertised rank. If its link quality is nearly perfect, it will stick to it. Otherwise, a few losses will happen, resulting in a increase of the link ETX. Then, a new neighbor will be tried out, until its link ETX worsens, etc. Overall, the difference with traditional CTP or RPL is that, instead of preserving an acceptable next hop, we keep switching among potentially good forwarders.

B. Cache Management

A second challenge comes from sensor motes having limited memory capacity, typically a few kilo-bytes of RAM, which is mostly used for the OS, the protocol stack and applications, leaving little space to neighbor information. For example, existing RPL implementations on the Tmote Sky platform have reported maximum cache sizes around 30 neighbors [8]. With expected densities of hundreds of nodes, the operating system has to choose which neighbors to keep track of. A simple solution such as the one used today in Contiki is to use a node lifetime policy, where the neighbors maitained in cache are the most recently used or heard from. We argue that this policy is a good option in highly dynamic networks, where neighbor information (both link estimation and rank) quickly perish. The downside of this policy is that there is excessive churn on alternate parents, and each will need to be re-evaluated each time it re-integrates the table.

Proposed Solution: To reduce unecessary churn on alternate neighbor entries, we propose two cache management alternatives.

a) Best Path Cost: This policy consists in filling the cache with the set of neighbors having the best path ETX. When receiving a new DIO, if the rank advertised is good enough, the neighbor will integrate the table. Otherwise, it is not considered at all. This strategy, by keeping information on good neighbors, allows for quick recovery in case a parent

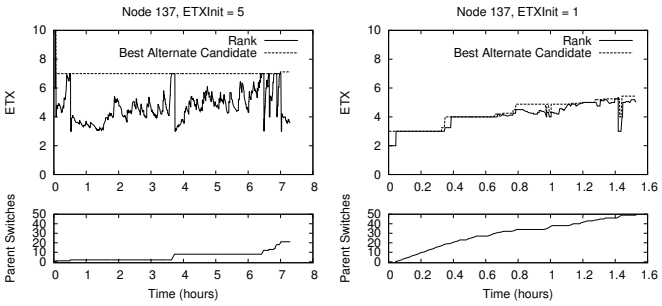


Fig. 2. Neighbor cache evolution in case of a poorly-connected node. With an init ETX of 5.0 (left), the link quality of the preferred parent varies highly, but few parent switches occur. With an init ETX of 1.0 (right), the passive probing mechanism comes into play, allowing to investigate each good neighbor through many parent switches.

switch is needed. However, our experience on testbeds shows that this cache management technique suffers from deleted neighbor rediscovery. We hypothesize that another limitation is the supression of good but non-perfect links in case of a large number of new nodes appearing at once.

b) Worst Link Estimation: This policy consists in keeping information about the neighbors with the worst link quality estimates. It may be needed when passive probing is used, because passive probing bases on optimistic link estimation. By keeping information about bad links, we avoid re-evaluating them too often.

IV. Preliminary Results

A. Setup

We use a pre-release of Contiki 2.6 which provides ContikiRPL, designed to operate in conjunction with Contiki's IPv6 stack and underlying MAC and Radio Duty Cycling protocols [13]. Contiki is a modular operating system with several isolated neighbor tables – our implementation adds logical interfaces between ContikiRPL, the IPv6 stack and the neighbor management module in order to acheive an efficient operation in bounded memory. We use the *Best Path Cost* policy discussed in Section III-B.

We run our experiments on the TU-Berlin TWIST testbed [5], which features 100 TelosB motes spread over a three-floor office building. The average node-density at default radio power is roughly 40.

We consider a simple data-collection application where each node in the network periodically sends a UDP message to the sink node. We use an inter-packet interval of one minute, with a random jitter. We use Contiki's uIPv6 network stack over a CSMA Mac protocol, without radio duty-cycling. This allows us to focus on RPL rather than the effects of duty cycling.

B. Link Metric Initialization

Our first series of experiments outlines some effects of link metric initialization. Figure 2 shows the evolution of the rank and would-be rank through the best alternate parent of a certain outlying poorly-connected node, at cache size of 30. Parent switches occur when the preferred parent total path cost rises above the best neighbor's total path cost beyond the hysteresis threshold of MHROF, which we have reduced to a small value.

On the left, at initial ETX of 5.0 (current default in ContikiRPL), we observe a high tolerance to losses for the reason depicted in Figure 1. The same parent, although providing a poor link ETX, remains preferred over 30 mostly untested neighbors during long periods until extreme and successive losses occur.

On the right, we show the detailed initial behavior of the system when using passive probing by lowering the initial link ETX to 1. The entire cache is tested at first, followed by a gradual increase of best alternate neighbors' ETX after probing each neighbor at least once. Sudden drops in rank occur regularly using this technique, as new neighbors are added to the cache with biased $ETXInit = 1$, then quickly tested and brought closer to their real link ETX. We describe this behavior as passive probing, in the sense that it does not require specific probing mechanisms or any additional RPL control messages. In high-density networks, this agile neighbor evaluation favors the detection of good-quality neighbors in the presence of a high number of poorly-connected parents.

C. Cache Size Control

Figure 3 shows the occurrences of preferred parent switches for a poorly-connected node on the TWIST tested, for three different cache sizes and $ETXInit = 1$. The corresponding transmission count per packet is given in Figure 4. Cache sizes 2 and 8, respectively 20 and 5 times lower than the average density on the TWIST testbed, exhibit parent switching rates equal to the data packet sending rate of 60 per hour, over the full 2-hour test period.

At a cache size of 2, the poorly-connected node fails to conserve the least amount of useful information on the neighborhood because the least retransmission or packet drop will favour the alternate, untested parent in a cyclic manner.

Increasing the cache size to 8 doesn't help improving the situation. The cache is still too small to store information on each useful (from a RPL perspective) neighbor. Therefore, RPL keeps looping on each neighbor, periodically re-evaluating their link quality.

When increasing the cache size to 30, we successfully capture the few good-quality neighbors available through the passive probing mechanism. At bootstrap, we observe 1 parent switch per data message, which is agile passive probing. Over time, the slope starts to decrease progressively as favourable neighbors are more frequently used than poor quality ones. However, our 1.5 hour experiments were not long enough to observe the convergence of the passive probing mechanism.

V. Related Work

To the best of our knowledge, density-scalable link estimation and neighbor management has not been directly addressed in the state of the art.

The Four-Bit link estimator [2] uses information from the physical, link and network layers to build link estimations. This allows the estimator to know, for example, if a link is

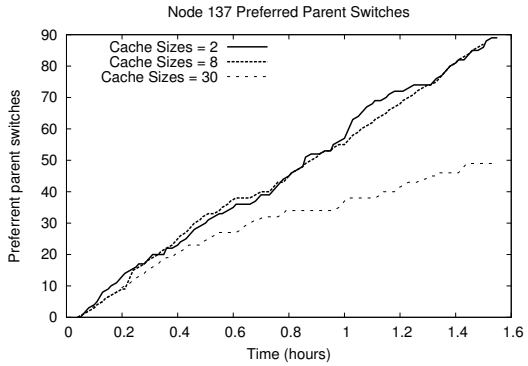


Fig. 3. Passive probing applied to a poorly-connected node with variable cache sizes. At bootstrap, all table sizes experience a parent switch rate of 1 switch per data message as the neighborhood is explored. At high cache sizes, we observe a gradual decrease in the parent switch rate as a few intermediate quality neighbors are found and favored. Low cache sizes suffer from the replacement of medium links by untested ones.

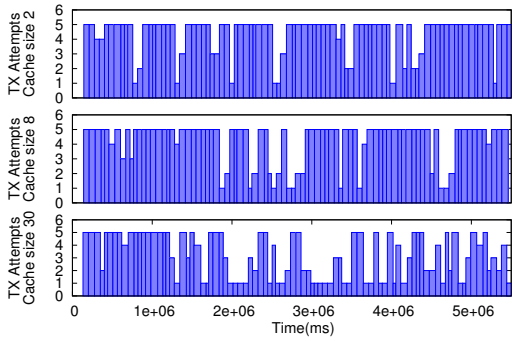


Fig. 4. Transmission count per application message for the same trials as Figure 3. Passive probing at large cache sizes explores more neighbors and favors good-quality links.

used for routing or not, and if a link is promising from a routing or application point of view. Our experience confirms that these information are of great importance.

[4] proposes implementation guidelines for RPL. The authors encourage route flapping for applications such as simple data collection over UDP. The passive probing mechanism proposed in this work follows these guidelines.

ORW [9] applies opportunistic routing to radio duty-cycled WSN, through a new metric, Expected Duty Cycles (EDC). ORW is particularly well-suited for dense networks, as it makes use of any next hop providing enough routing progress.

VI. Conclusion and Future Work

Achieving scalable routing is a great challenge for the IoT. Our study shows that scaling with density is particularly challenging, because of complex interactions between different components of the system: link quality estimation, cache management, routing parameters, etc. In fact, high density and steady-state operations with unknown links raises an extra challenge for in-testbed routing protocol evaluations, because state-of-the-art evaluation methodologies rely on using link PRRs to obtain ground truth [12].

Our results show that it is possible to achieve a probing-like behavior of RPL without sending specific probing control packets to alternate parents in the neighbor cache. This solution – passive probing – is however sensitive to cache size and requires to consider cache management policies when setting up the routing protocol.

In future work, we plan to investigate further different cache replacement policies and the trade-offs of active vs. passive probing.

Acknowledgment

This work was partly funded by the European Commission under FP7 with contract number FP7-ICT-2011.1.3-288879 and the Walloon Region under the First-DoCA funding number 1017211. The authors would like to thank Vlado Handziski from TU-Berlin for offering open access to the TWIST WSN testbed.

References

[1] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. In *Proceedings of the International Conference on Mobile Computing and Networking (ACM MobiCom)*, pages 134–146, San Diego, CA, USA, 2003. ACM.

[2] R. Fonseca, O. Gnawali, K. Jamieson, and P. Levis. Four-bit wireless link estimation. In *Proceedings of the Workshop on Hot Topics in Networks (ACM HotNets)*, Atlanta, Georgia, USA, Nov. 2007.

[3] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Berkeley, CA, USA, 2009.

[4] O. Gnawali and P. Levis. Recommendations for efficient implementation of rpl, Mar. 2012.

[5] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. TWIST: a scalable and reconfigurable testbed for wireless indoor experiments with sensor networks. In *Proceedings of the 2nd international workshop on Multi-hop ad hoc networks: from theory to reality (REALMAN'06)*, 2006.

[6] E. Kim, D. Kaspar, C. Gomez, and C. Bormann. Problem statement and requirements for 6lowpan routing, Mar. 2012.

[7] J. Ko, J. Eriksson, N. Tsiftes, S. Dawson-Haggerty, M. Durvy, J. Vasseur, A. Terzis, A. Dunkels, and D. Culler. Beyond Interoperability: Pushing the Performance of Sensornet IP Stacks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys)*, Seattle, WA, USA, Nov. 2011.

[8] J. G. Ko, O. Gnawali, D. Culler, and A. Terzis. Evaluating the performance of rpl and 6lowpan in tinyos. *Methodology*, 2011.

[9] O. Landsiedel, E. Ghadimi, S. Duquennoy, and M. Johansson. Low Power, Low Delay: Opportunistic Routing meets Duty Cycling. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM IPSN 2012)*, Beijing, China, Apr. 2012.

[10] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: A self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Mar. 2004.

[11] D. Popa. Applicability Statement for the Routing Protocol for Low Power and Lossy Networks (RPL) in AMI Networks, May 2012. draft-ietf-roll-applicability-ami-06, Internet-draft, work in progress.

[12] D. Puccinelli, O. Gnawali, S. Yoon, S. Santini, U. Colesanti, S. Giordano, and L. Guibas. The impact of network topology on collection performance. In *Proceedings of the 8th European Conference on Wireless Sensor Networks (EWSN 2011)*, pages 17–32, Bonn, Germany, Feb. 2011.

[13] N. Tsiftes, J. Eriksson, and A. Dunkels. Low-Power Wireless IPv6 Routing with ContikiRPL. In *Proceedings of the International Conference on Information Processing in Sensor Networks (ACM/IEEE IPSN)*, Stockholm, Sweden, Apr. 2010.

[14] T. Winter (Ed.), P. Thubert (Ed.), and RPL Author Team. Rpl: Ipv6 routing protocol for low power and lossy networks, Mar. 2012.